Roku Analytics Component

Available since firmware version 8.

Table of Contents

Overview

Roku analytics component is a component library which implements Google Analytics, Omniture, and Ooyala Analytics, amongst others. The library creates a simple method for using SceneGraph channels with one or more of these analytics solutions.

Support Model

The analytics platforms supported by the Roku Analytics component are classified into two different models:

Model #1

The channel passes video related events to the analytics module as they are received, and the analytics module is responsible for building and firing any analytics beacons needed for that event. This approach is most common for services focused only on video metrics. Platforms that use this model include **Brightcove** and **Ooyala**.

Model #2

The channel is responsible for deciding when an analytics beacon needs to be fired and collecting the data that needs to sent with the beacon. The analytics module uses that data to build and fire the beacon. This approach is most common for services that track UX interactions as well as video metrics. Platforms that use this model include **Omniture** and **Google Analytics**.

Supported Vendors

The Roku Analytics component officially supports the following vendors. Other vendors may be partially supported but may need to be updated to be compatible with the Roku Analytics component.

Ooyala

Model #1

Initialization Attributes

Attribute	Type	Required	Description
pcode	String	Required	Ooyala publisher code
userinfo	roAssociativeArray of Strings	Optional	User information
geoinfo	roAssociativeArray of Strings	Optional	Geographic information

Vendor-specific attributes for `setContentMetadata`

Attribute	Type	Required	Description
assetType	String	Required	Possible values include "ooyala" or "external"
assetID	Integer	Optional	The video's embed code - If omitted, the ID attribute from the standard metadata is used
duration	Integer	Optional	The duration of the video - If omitted, the length attribute from the standard metadata is used

Brightcove

Model #1

Initialization Attributes

Attribute	Type	Required	Description
account	String	Required	Brightcove account ID
user	String	Optional	Unique ID for this user

Vendor-specific attributes for `setContentMetadata`

Attribute	Type	Required	Description
source	String	Optional	If omitted, the `url` attribute from the standard content metadata is used
destination	String	Optional	If omitted, the `url` attribute from the standard content metadata is used
video	Integer	Optional	If omitted, the `id` attribute from the standard content metadata is used
video_name	String	Optional	If omitted, the `title` attribute from the standard content metadata is used
video_duration	Integer	Optional	If omitted, the `length` attribute from the standard content metadata is used. If length is also absent, the `duration` field from the Video Node is used

Omniture

Model #2

Attribute	Type	Required	Description

baseURL	String	Required	The URL of the Omniture suite that data should be sent to
persitentParams	roAssociativeArray	Optional	A set of static parameters and values that should be sent with every request

Analytics vendors using Model #2 use `trackEvent` rather than `setContentMetadata`.

Google Analytics

Model #2

Attribute	Type	Required	Description
trackingID	String	Required	The ID of the Google Analytics property that data should be sent to
defaultParams	roAssociativeArray	Optional	A set of static parameters and values that should be sent with every request

To use the Roku Analytics Component with Google Analytics, select the "Website" property view when you set up the Google Analytics dashboard; name your Website property accordingly.

Please make sure to note your trackingID from the Google Analytics Admin Dashboard.

Analytics vendors using Model #2 use `trackEvent` rather than `setContentMetadata`.

Google Analytics attributes for `trackEvent` can be found at https://developers.google.com/analytics/devguides/collection/protocol/v1/parameters

Guidelines

Manifest entry to use the Roku Analytics component library:

```
sg_component_libs_required=Roku_Analytics
```

Video Node - `notificationInterval`: For vendors using model #1, the `notificationInterval` field of the Video Node must be set to `1`. The component will check this when setting `initVideoPlayer` and change the value if necessary.

Mid-roll ads: When using multiple Video Nodes for client-side inserted mid-roll ads, `initVideoPlayer` must be set for each Video Node that is created. However, `finishedVideoPlayback` should only be set **once** at the end of the content session. `finishedVideoPlayback` **should not be set** at the end of mid-roll ads.

Implementation

To use the Roku Analytics component, add a field, "RSG_analytics," to `m.global` and then create an roSGNode object like so:

```
m.global.addField("RSG_analytics","node",false)
m.global.RSG_analytics = CreateObject("roSGNode","Roku_Analytics:AnalyticsNode")
```

`addField` takes three parameters:

- The `fieldName` - the name of the field to add
- The `type` of the field to add
- And an `alwaysNotify` value which determines if observers are triggered when the field changes or when the value of the field is set

Initialization

This method takes a roAssociativeArray of roAssociativeArrays containing configuration data for each analytics service such as endpoint URLs, API keys, etc. See Supported Vendors for vendor-specific configuration data.

Example:

```
m.global.RSG_analytics.init = {
    IQ : {
        PCODE : "pcode_value"
    }
    google : {
        trackingID : "trackingid_value"
        ' For convenience, this allows developers to define a set of parameters and values that will be
sent with every omniture call
        defaultParams : {}
        ' These values will be populated automatically by the component but can be overridden if desired
        an : “app_name”
        av : “app_version”
        cid : “client_id”
        sr : “screen_resolution” 'must be expressed as WWWWxHHHH
    }
    omniture : {
        baseURL : “https://omniture.suite.url/”
        ' For convenience, this allows developers to define a set of parameters and values that will be
sent with every omniture call
        defaultParams : {}
    }
}
```

Methods

Model #1

initVideoPlayer

> *This method can only be used for vendors using model #1 such as Ooyala or Brightcove.*

This method takes a single roAssociativeArray with exactly one attribute named `video` containing a SceneGraph Video Node. If your channel uses multiple Video Nodes (as might be done for mid-roll ads), this method needs to be set each time a new Video Node is created.

Example:

```
m.global.RSG_analytics.initVideoPlayer = {
    video: m.video
}
```

setContentMetadata

> *This method can only be used for vendors using model #1 such as Ooyala or Brightcove.*

This method takes a roAssociativeArray of roAssociativeArrays. At least one sub-AA is required and must contain the content meta-data for playback. Any other sub-AAs may contain additional information required for analytics providers and are optional.

Example with only Roku content meta-data:

```
myContent = {
    streamFormat = "mp4"
    streamUrl = "www.mycontent.com/video.mp4"
}

m.global.RSG_analytics.setContentMetadata = {
    content: myContent
}
```

Example with Roku content meta-data and additional analytics provider information:

```
myContent = {
    streamFormat = "mp4"
    streamUrl = "www.mycontent.com/video.mp4"
}

metadata = {
    duration : item.length,
    assetId : item.id,
    assetType : "external"
}

m.global.RSG_analytics.setContentMetadata = {
    content: myContent
    IQ : metadata
}
```

finishedVideoPlayback

> *This method can only be used for vendors using model #1 such as Ooyala or Brightcove.*

This method is similar to `initVideoPlayer` and takes a single roAssociativeArray with exactly one attribute named `video` containing a Scenegraph Video Node. This should be set once video playback has finished which will allow the component to finish analytics tasks and stop observing Video Node events.

Unlike `initVideoPlayer`, this method should only be set once the last Video Node is closed (i.e., do not set this when closing a Video Node after a mid-roll ad).

Example:

```
sub onVideoState()
    closeStates = {
        finished : "",
        error : ""
    }
    if closeStates[m.video.state] <> invalid then
        'Send video player so analytics node could unobserve all fields and close session properly
        m.global.RSG_analytics.finishedVideoPlayback = {
            video: m.video
        }
        hideVideo() 'implement this to restore prev screen
    end if
end Sub
```

Model #2

trackEvent

> *This method can only be used for vendors using model #2 such as Google Analytics or Omniture.*

This method takes a roAssociativeArray of roAssociativeArrays containing the parameters and values that are sent with a beacon. This method can be set anytime an event needs to be fired.

> *Google Analytics attributes*
>
> *Omniture attributes*

Example:

```
m.global.RSG_analytics.trackEvent = {
    google: {
        ec: “navigation”,
        ea: “channel_launch”
    },
    omniture: {
        events: “event15,event17”,
        page_name: “splash_screen”,
        c17: “channel_launch”
    }
}
```

Debug

This method takes a boolean to determine if debug data will be shown in the console. The default value is `false` (No debug data will be displayed in the console).

Example:

```
m.global.RSG_analytics.debug = true
```

Please note that the library **does** send beacons even in debug mode.

Example

Following is a simple example of using the analytics component with a service that supports Model #1.

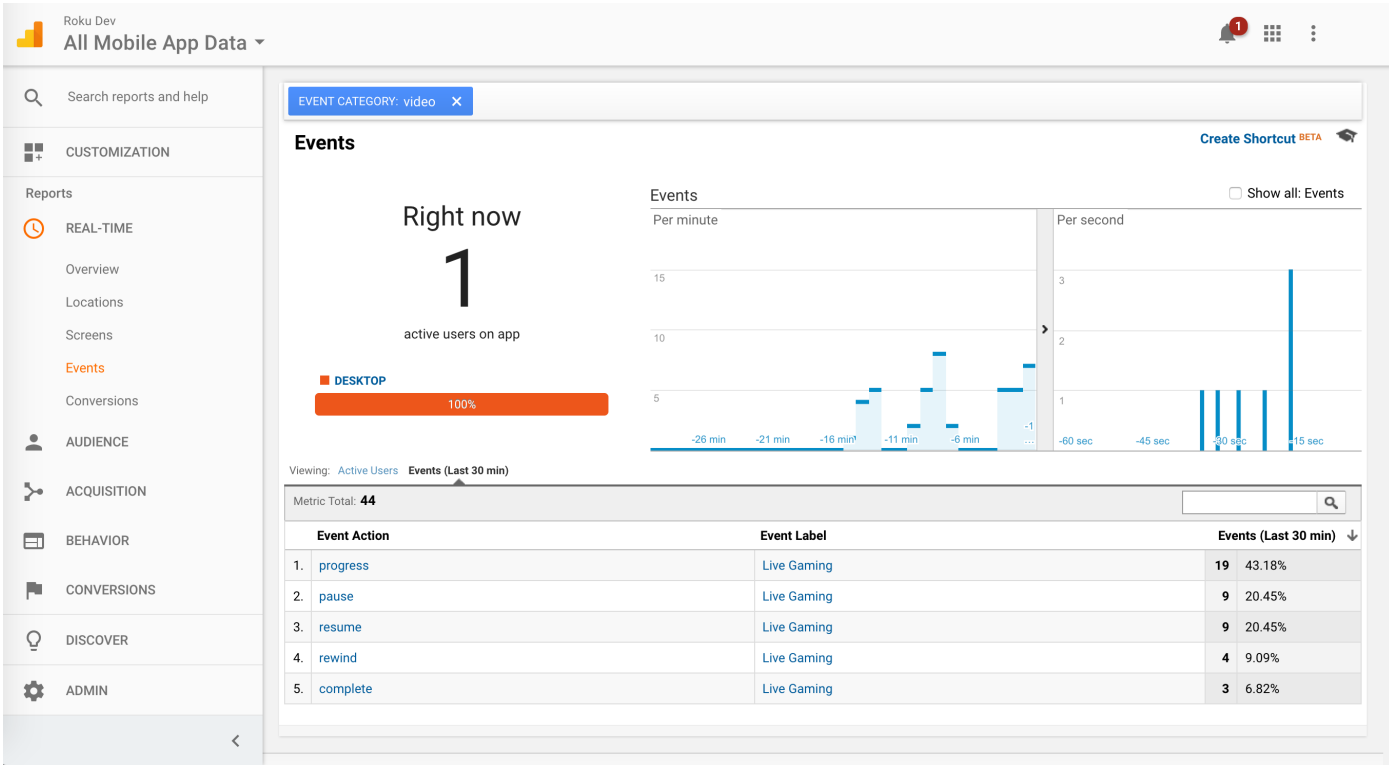
```
Sub VerySimpleShowVideo(item)
    m.global.addField("RSG_analytics","node",false)
    m.global.RSG_analytics = CreateObject("roSGNode","Roku_Analytics:AnalyticsNode")

    ' Analytics Initialization
    m.global.RSG_analytics.init = {
        IQ : {
            PCODE : "pcode_value"
        }
    }

    m.video = m.top.createchild("roSGNode","Video")
    m.video.notificationInterval = 1
    m.video.content = item

    'Setup analytics for this video player
    'We will pass IDs of analytics that should take part in this video playback
    m.global.RSG_analytics.initVideoPlayer = {
        video: m.video
    }
    'set IQ specific metadata
    'This metadata is IQ specific and cannot be stored in ContentNode
    metadata = {
        duration : item.length,
        assetId : item.id,
        assetType : "external"
    }
    m.global.RSG_analytics.setContentMetadata = {
        IQ : metadata
    }

    m.video.observeField("state","onVideoState")
    m.video.control = "start"
End Sub

Sub onVideoState()
    closeStates = {
        finished : "",
        error : ""
    }
    if closeStates[m.video.state] <> invalid then
        'Send video player so analytics node could unobserve all fields and close session properly
        m.global.RSG_analytics.finishedVideoPlayback = {
            video: m.video
        }
        hideVideo() 'implement this to restore prev screen
    end if
end Sub
```

Expected Output

Once you run the Roku Analytics Component, you should see the beacons it fires on the BrightScript console. The analytics are displayed on your Analytics dashboard.

Example:

Google Analytics Dashboard after running Analytics Component:



Sample Channels

To test the library, side-load the sample channels below with respect to the Analytics platform you are using. You can test either Ooyala (Model #1) or Google Analytics (Model #2).

Use the Sample channel #1 to test Ooyala Analytics. Change the publisher code (PCODE) in the baseScene.brs to your Ooyala account PCODE to see the analytics.

When using the sample channel for Google Analytics (Model #2), make sure to change the tracking ID to your Google Analytics tracking ID in baseScene.brs under the folder components.

Model	Download
Simple model #1	RokuAnalyticsComponent_Model1.zip
Simple model #2	RokuAnalyticsComponent_Model2.zip